Circuit Clustering for Delay Minimization under Area and Pin Constraints*

Honghua Yang and D. F. Wong

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712

Abstract

We consider the problem of circuit partitioning for multiple-chip implementation. One motivation for studying this problem is the current needs of good partitioning tools for implementing a circuit on multiple FPGA chips. We allow duplication of logic gates as it would reduce circuit delay. Circuit partitioning with duplication of logic gates is also called circuit clustering. In this paper, we present a circuit clustering algorithm that minimizes circuit delay subject to both area and pin constraints on each chip, using the general delay model. We develop a repeated network cut technique for finding a cluster that is bounded by both area and pin constraints. Our algorithm achieves optimal delay under either the area constraint only or the pin constraint only. Under both area and pin constraints, our algorithm achieves optimal delay in most cases. We outline the condition under which the non-optimality occurs and show that the condition occurs rarely in practice. We tested our algorithm on a set of benchmark circuits and consistently obtained optimal or near-optimal delays.

1 Introduction

VLSI circuit partitioning [PL88] underlies many important problems in VLSI layout designs, such as floorplanning and placement, as well as mapping a complicated system onto multiple-chip modules, where the partitioning must satisfy a set of design constraints such as area constraints, pin constraints, timing constraints, and thermal constraints. In this paper, we consider the problem of partitioning a circuit into chips to minimize circuit delay subject to both area and pin constraints on each chip. We allow duplication of logic gates as it would reduce circuit delay. Circuit partitioning with duplication of logic gates is also referred to as *circuit clustering*.

Most previous research on circuit clustering for delay minimization focused on either area-constrained or pin-constrained clustering, but not both. Lawler, Levitt and Turner [LLT66] presented a polynomial time delay optimal algorithm for area-constrained circuit clustering assuming a *unit delay model*, in which the only delay accounted for is a constant delay on each inter-chip wire. Cong and Ding [CD92] presented a delay optimal technology mapping algorithm for lookup-table based FPGA designs. It can be shown that their algorithm in fact is a delay optimal algorithm for pin-constrained clustering under the unit delay model. The unit delay model is not very realistic. In general, a signal path may pass through many gates inside a chip, hence incurring a substantial delay within the chip itself. Murgai, Brayton and Sangiovanni-Vincentelli [MBSV91] proposed a *general delay model*. In this model, we have (1) a *gate delay* $\delta(v)$ for each gate v; (2) no delay for any interconnection within a chip; (3) a constant delay of D units for every inter-chip interconnection. The authors of [MBSV91] presented a greedy algorithm for area-constrained clustering using the general delay model. Rajaraman and Wong [RW93] presented a delay optimal algorithm for the same problem. A major shortcoming of the algorithms in [MBSV91] and [RW93] is that they do not consider pin constraints.

Delay minimization under both area and pin constraints is a difficult problem because of the totally different nature of the two types of constraints (e.g. area constraints is monotone and pin constraints is not [LLT66]). In this paper, we present an efficient circuit clustering algorithm to minimize delay using the general delay model, subject to both area and pin constraints. We develop a repeated network cut technique to find a cluster that is bounded by both area and pin constraints. Our algorithm achieves optimal delay under either the area constraint only or the pin constraint only. Under both area and pin constraints, our algorithm is optimal in most cases. We outline the condition in which the non-optimality occurs and show that the condition occurs rarely in practice. We tested our algorithm on a set of iscas85 and MCNC benchmark circuits and consistently obtained optimal or near-optimal delays. When compared with the classic FM [FM82] partitioning heuristic without gate duplication, our algorithm achieves significant improvement in delays while using comparable number of chips.

The rest of the paper is organized as follows. In section 2, we formally define our problem. Section 3 presents an algorithm for solving the problem. The algorithm relies on a procedure that finds an area/pin-constrained sub-circuit rooted at a particular node. Section 4 describes a repeated network maximum flow technique to implement such procedure, and analyzes the complexity of our algorithm. Section 5 discusses

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459, by an Intel Foundation Graduate Fellowship, by a DAC Design Automation Scholarship, and by a grant from AT&T Bell Laboratories.

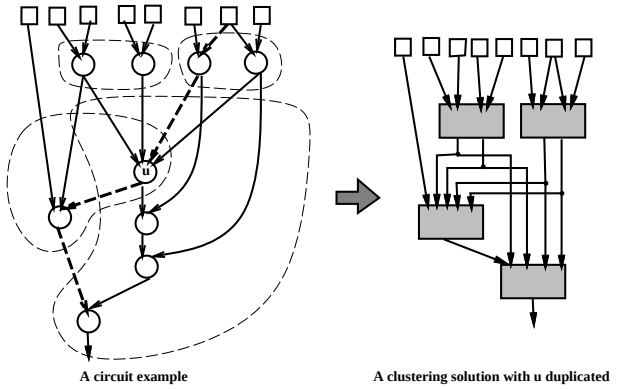

Figure 1: An example of circuit clustering with $A = 4$, $P = 6$. Each circled area is a cluster.

the post-processing step of our algorithm that reduces gate duplication and the number of clusters without increasing circuit delay. Finally we show some experimental results in section 6 and conclude the paper in section 7.

2 Problem Formulation

A *combinational circuit* is represented by a directed acyclic graph (DAG) $N = (V, E)$ where V is the set of nodes representing gates[1], and E is the set of directed edges representing interconnections between gates. Each node v in N is assigned a positive *weight* $w(v)$. A *cluster* is a subset of nodes in V. A *solution* to the *circuit clustering problem* of N is a logically equivalent circuit N_C with a node partition such that (1) N_C consists of N with some replicated nodes and edges, (2) each set in the node partition is a cluster in N, and (3) a primary input (PI) node is not included in any cluster.

An *inter-cluster edge* is an edge in N_C that connects gates in two clusters. The *inter-cluster delay* incurred at an inter-cluster edge is denoted by a constant D. The *input (output) nodes of a cluster* C in N_C, denoted by $Input(C)$ ($Output(C)$), are the nodes outside of C that have wires incoming to (outgoing from) nodes in C. The area of C, denoted by $Area(C)$, is the sum of the gate weights in C. Let A and P be two given constants denoting the *area constraint* and the *pin constraint*, respectively. A cluster C in a solution is *feasible* if $Area(C) \leq A$, and $|Input(C)| + |Output(C)| \leq P$. A solution N_C is *feasible* if every cluster in N_C is feasible. Given a solution N_C, the *delay along a path* in N_C is the sum of the gate delays and inter-cluster delays on the path. The *delay of a node* v in N_C is the maximum delay along any path from a PI node to v. The *delay through the circuit* is the maximum delay among all primary output (PO) nodes. A feasible solution N_C is *(delay) optimal* if the delay through N_C is the minimum among all feasible solutions.

Figure 1 shows an example of circuit clustering, where node u is duplicated. If $D = 2$, and $\delta(v) = 1$ for all gate v, then the path represented by the dashed edges is a critical path of total delay 8 since it contains 2 inter-cluster delays and 4 gate delays.

We use the following notation throughout the paper. Given a node v in a circuit N, N_v denotes the subgraph of N induced by v and all its predecessors, with v being the only output node; a cluster *rooted at* v, denoted by C_v, is a subgraph of N_v containing v as the only output node. Let $d(v)$ denote the delay of a node v in an optimal clustering solution of N_v. When computing a feasible cluster for a node v, it suffices to consider the sub-circuit N_v consisting of v and all its predecessors, since $d(v)$ depends only on the predecessor of v. Thus, we can replace the pin constraint of a feasible rooted cluster C_v by $|Input(C_v)| \leq P - 1$.

3 An Optimal Clustering Algorithm

Similar to the optimal algorithm given in [RW93], our algorithm CLUSTER contains two phases: *labeling* and *clustering*. In the labeling phase, for each node v in N in topological order, we compute a cluster C_v which would give the optimal delay $d(v)$ for v. In the clustering phase, clusters are generated starting from the PO nodes in a bottom-up fashion, based on the clusters obtained in the labeling phase.

We now describe the computation of C_v and $d(v)$ for a node v in the labeling phase, assuming $d(u)$ for all predecessors u of v in N_v have already been computed since u precedes v in the topological ordering. Let $\Delta(u, v)$ denote the maximum total *gate delay* along any path in N from the output of u to the output of v, and let $d'(u, v) = d(u) + \Delta(u, v)$, for any $u \in N_v$ ($u \neq v$). Given a cluster C_v for v, $d(v) = \max\{d'(u, v) + D \mid u \in Input(C_v)\}$. In order to minimize $d(v)$, the greater the value of $d'(u, v)$, the more need to include u in C_v. Hence we sort different values of $d'(u, v)$ into decreasing order:

$$d'_1 > d'_2 > \ldots > d'_{k-1} > d'_k > d'_{k+1} > \ldots, \quad \text{where}$$

$$\sum_{d'(u,v) \geq d'_k} w(u) \leq A, \quad \text{and} \quad \sum_{d'(u,v) \geq d'_{k+1}} w(u) > A.$$

Let $C^l = \{u \mid d'(u, v) \geq d'_l\}$. Initially, we set l to k, and C^k is the set of the highest d'-valued nodes u that can fit in a cluster without violating the area constraint[2]. If we can find a feasible C_v containing C^l in step 10, then we assign $d(v)$ the largest d' value outside C_v plus the inter-cluster delay D in step 12, and hence $d(v) = d'_{l+1} + D$. Otherwise, we remove the smallest d'-valued nodes from C^l (in step 13). This process is repeated until C_v is found. A labeling process of node v is shown in Figure 2. The CLUSTER algorithm is listed in Algorithm 1.

Theorem 3.1 *The algorithm CLUSTER finds a delay optimal solution for the circuit clustering problem provided that step 10 always finds a feasible C_v when there exists one.*

[1] A gate is either a simple gate or a complex gate such as a lookup-table in an FPGA.

[2] The algorithm in [RW93] would return C^k as an optimal cluster for v since it is not concerned with pin constraint, while steps 8-13 in our algorithm CLUSTER try to find an optimal cluster for v that satisfies both area and pin constraints.

Algorithm 1 CLUSTER
Input: A combinational circuit $N = (V, E)$, A, P and D.
Output: A clustering solution of N and its delay.
begin
/* The labeling phase */
0. Assign 0 to $d(u)$ for each PI node u;
1. Sort the non-PI nodes in V by topological order into a list V_T;
2. **for** each node v in V_T in topological order
3. Construct N_v;
4. Calculate $\Delta(u, v)$ for every $u \in N_v$ to v;
5. Calculate $d'(u, v) = d(u) + \Delta(u, v)$ for every $u \in N_v$ except v;
6. Sort all different values of $d'(u, v)$ in strictly decreasing order: $d'_1 > d'_2 \ldots > d'_m$;
7. Let $k = \max\{i \mid$ the nodes u with $d'(u, v) \geq d'_i$ have area $\leq A\}$;
Let $C^l = \{u \mid d'(u, v) \geq d'_l\}$;
8. $done = false$; $l = k$;
9. **while** *not done*
10. Find a feasible cluster C_v containing all nodes in C^l;
11. **if** such C_v exists
12. $d(v) = d'_{l+1} + D$; $done = true$;
13. **else** $l = l - 1$;
/* The clustering phase */
14. $S = \phi$; $L =$ the list of PO nodes in N;
15. **while** L is not empty
16. Remove a node v from L;
17. $S = S \cup \{C_v\}$; $L = L \cup Input(C_v)$.

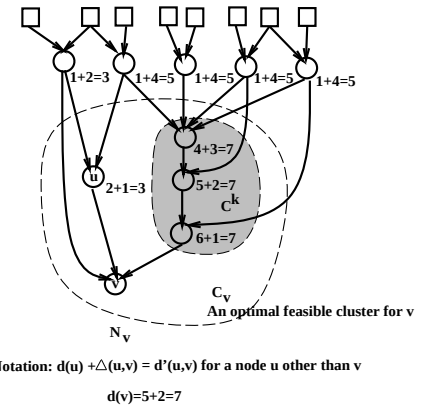

Figure 2: A labeling process for node v, assuming $A = 5$, $P = 6$, $D = 2$, $w(v) = 1, \delta(v) = 1$ for each node v.

Clearly, the optimality of the solutions produced by our algorithm depends on how to implement step 10. An exact implementation would be to enumerate every possible node set of $\leq P - 1$ nodes, and test if it constitutes the input nodes of a cluster rooted at v. Although this implementation has a polynomial time complexity $O(|N|^{P+c})$ for some nonnegative constant c, it is far from realistic since the constant P is usually very large. Therefore, in the next section, we will instead propose an efficient algorithm using network flow techniques to implement step 10 that runs in $O(P^3 mn \log n)$ time. The algorithm finds a feasible cluster rooted at v containing C^k in most cases when there indeed exists one. Consequently, our algorithm always finds optimal or near optimal solutions to the circuit clustering problem.

4 Finding a Feasible C_v Containing C^k

To find a cluster containing C^k, we collapse all nodes in C^k to node v and add weights of these collapsed nodes to the weight of v (see Figure 3). We then find a feasible cluster rooted at v by repeatedly

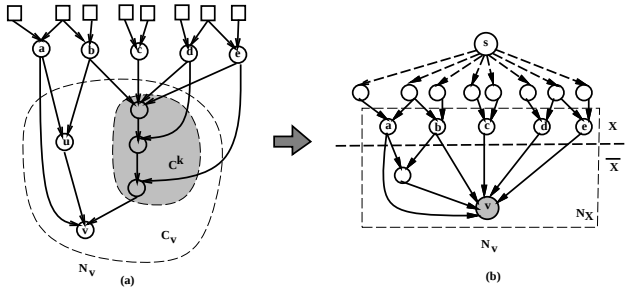

Figure 3: Collapsing C^k to v.

applying max-flow min-cut techniques.

We first augment N_v to a single-source single-sink graph by identifying v as the sink and adding a source s with an edge going to each PI node in N_v. We define a *cut* $(X, \bar{X})$ to be a partition of nodes in N_v such that $s \in X$ and $v \in \bar{X}$. The edge cut $e(X, \bar{X})$ is the set of edges in N_v whose starting node is in X and ending node is in $\bar{X}$. The node cut $n(X, \bar{X})$ is the set of nodes in X which are the starting nodes of the edges in $e(X, \bar{X})$. Figure 3 (b) shows a cut $(X, \bar{X})$, where $n(X, \bar{X}) = \{a, b, c, d, e\}$. Note that $\bar{X}$ corresponds to C_v, and $n(X, \bar{X})$ corresponds to $Input(C_v)$.

A cut $(X, \bar{X})$ of N is a *min-cut*[3] if its induced node cut $n(X, \bar{X})$ contains the minimum number of nodes among all cuts of N, i.e. $|n(X, \bar{X})|$ is minimum. A min-cut $(X, \bar{X})$ of N is a *min-area min-cut* if $Area(\bar{X})$ is minimum among all min-cuts of N. A cut $(Y, \bar{Y})$ of N is an *undercut* of another cut $(X, \bar{X})$ of N if $|n(Y, \bar{Y})| \leq |n(X, \bar{X})|$ and $\bar{Y} \subseteq \bar{X}$. The *cut* $(X, \bar{X})$ *induced subnetwork*, denoted by N_X, is the subgraph of N induced by $n(X, \bar{X}) \cup \bar{X}$ (see Figure 3 (b)).

It is easy to show that a feasible cluster rooted at v corresponds to a cut $(X, \bar{X})$ of N_v such that $Area(\bar{X}) \leq A$ and $|n(X, \bar{X})| \leq P - 1$. We call such a cut a *feasible cut*. Further, any undercut of a feasible cut is still a feasible cut.

Our algorithm works as follows. We first find a min-area min-cut $(X, \bar{X})$ in N_v. If $|n(X, \bar{X})| > P - 1$, then no feasible cluster rooted at v containing C^k can be found. If both $|n(X, \bar{X})| \leq P - 1$ and $Area(\bar{X}) \leq A$, then we return $\bar{X}$ as the feasible cluster C_v. If $|n(X, \bar{X})| \leq P - 1$ but $Area(\bar{X}) > A$, then we try to find in N_X a *min-area min-cut* $(X_u, \bar{X}_u)$ *that excludes at least one node* u *in* $n(X, \bar{X})$. Note that $(X_u, \bar{X}_u)$ has a larger node cut size than $(X, \bar{X})$, but $\bar{X}_u$ has a smaller area than $\bar{X}$. We designate $(X_u, \bar{X}_u)$ as $(X, \bar{X})$ and repeat the above procedure. The iteration stops when either a feasible cut is found or no new cut $(X, \bar{X})$ with $|n(X, \bar{X})| \leq P - 1$ can be found. Figure 4 illustrates the repeated cut process.

In Lemma 4.1, we show that our repeated cut process finds an optimal solution in most cases.

[3]Note that we define a min-cut as a cut with the minimum node cut size, while traditionally a min-cut is defined as a cut with the minimum edge cut size.

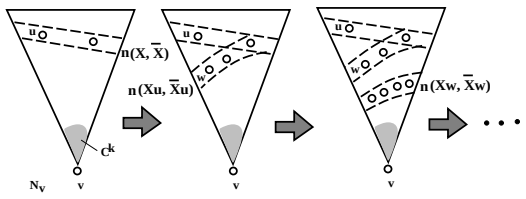

Figure 4: Illustration for the repeated cut process.

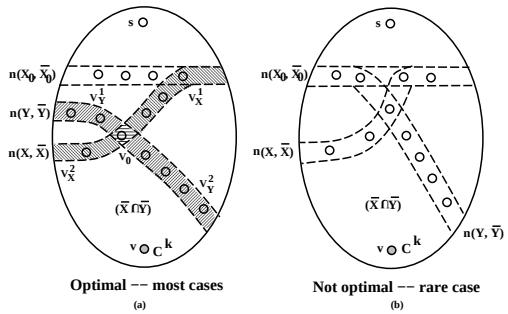

Figure 5: Illustration for Lemma 4.1.

Lemma 4.1 *Let $n(X_0, \bar{X}_0)$ be any node cut in N_v. Let $(X, \bar{X})$ be a min-area min-cut in N_{X_0} that excludes some node in $n(X_0, \bar{X}_0)$. Then any cut $(Y, \bar{Y})$ in N_{X_0} satisfying $n(X, \bar{X}) \cup n(Y, \bar{Y}) \not\supseteq n(X_0, \bar{X}_0)$ has an undercut $(X \cup Y, \bar{X} \cap \bar{Y})$ in N_X.*

Proof: We partition the nodes in $n(X, \bar{X})$ and $n(Y, \bar{Y})$ as follows: $V_0 = n(X, \bar{X}) \cap n(Y, \bar{Y})$, $V_X^1 = n(X, \bar{X}) \cap (Y - n(Y, \bar{Y}))$, $V_Y^1 = n(Y, \bar{Y}) \cap (X - n(X, \bar{X}))$, $V_X^2 = n(X, \bar{X}) \cap \bar{Y}$, and $V_Y^2 = n(Y, \bar{Y}) \cap \bar{X}$ (see Figure 5 (a)). Then the node cut $n(X \cap Y, \bar{X} \cup \bar{Y}) = V_X^1 \cup V_Y^1 \cup V_0$, and the node cut $n(X \cup Y, \bar{X} \cap \bar{Y}) = V_X^2 \cup V_Y^2 \cup V_0$. Since there is a node in $n(X_0, \bar{X}_0)$ that is not in $n(X, \bar{X}) \cup n(Y, \bar{Y})$, $n(X \cap Y, \bar{X} \cup \bar{Y})$ excludes at least one node in $n(X_0, \bar{X}_0)$. Since $(X, \bar{X})$ is a min-cut in N_{X_0} that excludes some node in $n(X_0, \bar{X}_0)$, we have $|n(X \cap Y, \bar{X} \cup \bar{Y})| \geq |n(X, \bar{X})|$, i.e., $|V_X^1| + |V_Y^1| + |V_0| \geq |V_X^1| + |V_X^2| + |V_0|$. By simplifying the equation, we have $|V_Y^1| \geq |V_X^2|$.

Hence, $|n(Y, \bar{Y})| = |V_Y^1| + |V_Y^2| + |V_0| \geq |V_X^2| + |V_Y^2| + |V_0| = |n(X \cup Y, \bar{X} \cap \bar{Y})|$. Further, $Area(\bar{Y}) \geq Area(\bar{X} \cap \bar{Y})$. Hence the cut $(X \cup Y, \bar{X} \cap \bar{Y})$ is an undercut of $(Y, \bar{Y})$ in N_X and the lemma holds. □

When searching for a feasible cut in N_{X_0}, we first find a min-area min-cut $(X, \bar{X})$ in N_{X_0} that excludes some node in $n(X_0, \bar{X}_0)$. Suppose the cut $(X, \bar{X})$ violates the area constraint (i.e., $Area(\bar{X}) > A$) only, but another cut $(Y, \bar{Y})$ in N_{X_0} satisfies both the area and pin constraint, we can narrow down our search for a feasible cut to the smaller circuit N_X to find an undercut of $(Y, \bar{Y})$ that is also feasible. The above lemma demonstrates that the only case our algorithm does not find a feasible cut in N_X while one exists in N_{X_0} is when (1) there exists a node u in $n(X_0, \bar{X}_0)$ and a min-area min-cut $(X, \bar{X})$ in N_{X_0} that excludes u, such that *for every* feasible cut $(Y, \bar{Y})$ in N_{X_0}, $n(X_0, \bar{X}_0) \subseteq n(X, \bar{X}) \cup n(Y, \bar{Y})$ holds; and (2) our algorithm is unlucky enough to pick the min-area min-cut $(X, \bar{X})$ in N_{X_0} that excludes u. (See Figure 5 (b).) Our experimental results confirm that this non-optimal condition occurs rarely in practice.

Note that we define our min-cut as a cut with the minimum *node cut size*. We apply the *node-splitting* technique ([Eve79, CD92]) to reduce the minimum node cut size constraint to the minimum edge cut size constraint. We construct N_v' from N_v as follows. For each node u in N_v except s and v, we define two nodes u_1 and u_2 with an edge (u_1, u_2) in N_v', which is called a *bridging edge*. For each edge (u, w) ($u \neq s$ and $w \neq v$) in N_v, we define an edge (u_2, w_1) in N_v'. For each edge (s, u) and (w, v) in N_v, we defined (s, u_1) and (w_2, v) in N_v' respectively. Then we assign unit capacity to all bridging edges and infinite capacity to all other edges in N_v'. For a pair of nodes u_1 and u_2 split from u, $w(u_1) = w(u)$, and $w(u_2) = 0$. The weight of v is not changed in N_v'.

Lemma 4.2 *N_v has a cut $(X, \bar{X})$ of node cut size K iff N_v' has a cut $(X', \bar{X}')$ of edge cut size K. Further, $Area(\bar{X}) = Area(\bar{X}')$.*

By the max-flow min-cut theorem [Eve79], the minimum capacity of an edge cut (i.e., the size of the edge cut, since a minimum capacity edge cut can have only bridging edges with unit capacity) is equal to a maximum total flow in N_v'. Since we are only interested in finding an edge cut with size no more than $P - 1$ in N_v', and each augmenting path in N_v' from s to v increases the flow by one unit (since a bridging edge has unit capacity), we only need to find at most P augmenting paths in N_v' to compute a maximum flow (also see [CD92]). If we can find P augmenting paths in N_v', then the maximum flow value and the minimum edge cut size in N_v' is at least P, and there is no need to compute the exact maximum flow in N_v'. Otherwise, we can find a cut $(X', \bar{X}')$ with the minimum edge cut size by letting $\bar{X}' = \{u |$ there exists an augmenting path from u to $v\}$, and let X' be the set of the remaining nodes in N_v'.

We can show that the cut $(X', \bar{X}')$ described as above is in fact a minimum edge cut with the minimum area in N_v'. By Lemma 4.2, a minimum edge cut with the minimum area in N_v' corresponds to a min-area min-cut in N_v.

Theorem 4.1 *Algorithm CLUSTER can be implemented to run in $O(P^3 mn \log(n))$ time where n, m are the numbers of nodes and edges in the circuit, respectively.*

Proof: The total number of iterations in steps 9-13 in algorithm CLUSTER is at most $\log(\min\{n, A\}) \leq \log(n)$ by using a binary search strategy. Each iteration takes $O(P^3 m)$ time. Since there are n nodes, the labeling phase takes $O(P^3 mn \log(n))$ time. □

Theorem 4.2 *By appropriately implementing step 10, the algorithm CLUSTER can always find a delay optimal solution for the circuit clustering problem under either the area constraint only or the pin constraint only.*

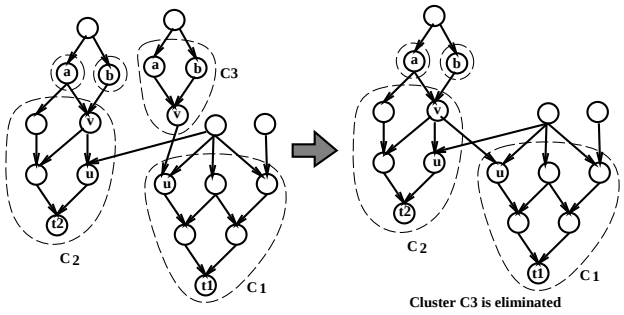

Figure 6: An example of rooted cluster elimination.

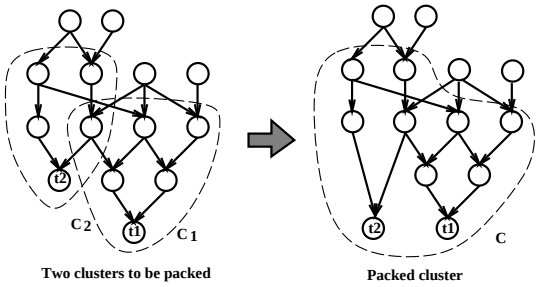

Figure 7: An example of cluster packing.

Proof: Under either condition, the implementation of step 10 of algorithm 1 is simplified. Under the area constraint only, step 10 simply returns C^k and our algorithm degenerates to the algorithm in [RW93], which runs in $O(n^2 \log(n) + mn)$ time.

Under the pin constraint only, step 10 only need find a single min-area min-cut, which takes $O(Pm)$ time. Hence the whole algorithm runs in $O(Pmn \log(n))$ time. □

Note that Theorem 4.2 gives the first polynomial time (delay) optimal algorithm for pin-constraint clustering under the *general* delay model. Previously, [CD92] gave a (delay) optimal algorithm for pin-constraint clustering under the *unit* delay model.

5 Post-Processing

After the clustering and generating phases in CLUSTER, the number of clusters in the resulting circuit is usually large, since many clusters are not to their full capacity and there are gate duplications. To find a feasible solution with the same optimal or near optimal delay computed in the labeling phase, but with less duplication and fewer number of clusters, we present the following simple two-step cluster reduction method: *rooted cluster elimination* and *cluster packing*. In the rooted cluster elimination step, we try to eliminate as many as possible rooted clusters by substituting the roots (i.e. the only output node) of these clusters with their replica in other clusters, provided that the pin constraint is not violated on any cluster, and that the delay of the clustered circuit computed by CLUSTER is not increased (see Figure 6). In the cluster packing step, we try to pack as many as possible clusters into a single cluster using a heuristic algorithm that favors the packing of two clusters sharing the largest number of gates (see Figure 7).

circuit	non-PI gates	CLUSTER-RW		CLUSTER	
		% of clu > 40 pins	opt del	del	del inc %
C499	202	66.7	13	13	0.0
C880	357	8.7	25	25	0.0
C1355	514	0.0	27	27	0.0
C1908	880	11.3	43	44	+2.3
C2670	1161	4.7	36	36	0.0
C3540	1667	18.7	50	51	+2.0
C5315	2290	6.2	53	53	0.0
C6288	2416	1.8	130	130	0.0
C7552	3466	5.5	46	47	+2.1
apex6	238	0.0	10	10	0.0
comp	55	0.0	8	8	0.0
cordic	102	0.0	15	15	0.0
dalu	1131	6.0	26	28	+7.7
des	681	0.0	6	6	0.0
pair	824	2.3	20	20	0.0
rot	138	10.9	11	13	+18.2
term1	147	0.0	11	11	0.0
mapped circuit	LUTs				
C499_map5	154	100.0	7	9	+28.6
C880_map5	233	12.9	11	12	+9.1
apex6_map	257	0.0	6	6	0.0
des_map5	1308	0.0	7	7	0.0
rot_map5	283	9.3	8	10	+25.0
comparison			1		+4.3%

Table 1: Comparison of delay obtained by CLUSTER-RW and CLUSTER under area constraint $A = 100$, pin constraint $P = 40$, inter-cluster delay $D = 2$, and $w(v) = 1, \delta(v) = 1$ for each non-PI node v.

6 Experimental Results

We implemented our CLUSTER algorithm, the CLUSTER-RW algorithm given in [RW93], and a partitioning algorithm (without gate duplication) based on the FM [FM82] heuristic, using the C language. We used input/output routines and general utility functions provided by MIS [BRSV87] in our implementation. We conducted testing on a set of iscas85 and MCNC benchmark combinational circuits, and a number of look-up table (LUT) based technology-mapped circuits obtained from Flow-Map [CD92]. We assumed that the inter-cluster delay $D = 2$. For ease of testing, we assigned unit gate weight and unit gate delay to each non-PI node (gate or LUT) in a circuit. Our algorithm and program can deal with non-uniform gate weight and gate delay.

We set the area constraint per cluster to $A = 100$ and tested the following (see Table 1): CLUSTER-RW with no pin constraint, recording the percentage of clusters that violate the pin constraint, and the optimal delay for area constraint only clustering; CLUSTER with $A = 100$ and pin constraint $P = 40$ per cluster, recording the delay and the percentage of increased delay compared with the optimal delay without the pin constraint.

Our results show that clustering without consideration of pin constraint, as in the case of CLUSTER-RW, often results in many clusters with unacceptably large pin numbers that violate the pin constraint. Our results also show that CLUSTER generates clusters that satisfy both area and pin constraints, and still consistently keeps the optimality or near optimality

circuit	non-PI gates	Area/Pin-FM		CLUSTER		Area/Pin-FM compared w/ CLUSTER (%)	
		#clu	del	#clu	del	#clu	del
C499	202	3	17	3	13	0.0	+30.8
C880	357	4	31	5	25	-20.0	+24.0
C1355	514	6	31	7	27	-14.3	+14.8
C1908	880	9	52	24	43	-62.5	+20.9
C3540	1667	95	81	51	50	+86.3	+62.0
C5315	2290	90	72	59	53	+52.5	+35.8
C6288	2416	25	159	65	130	-61.5	+22.3
C7552	3466	109	63	79	46	+38.0	+71.7
term1	147	2	12	2	11	0.0	+9.1
average						+2.1	+32.4

Table 2: Comparison of the number of clusters and the delay obtained by Area/Pin-FM and CLUSTER, under area constraint $A = 100$, pin constraint $P = 80$, inter-cluster delay $D = 2$, and $w(v) = 1, \delta(v) = 1$ for each non-PI node v.

of the delay of the circuit obtained in clustering with the same area constraint but without pin constraint. Note that even in the case that CLUSTER generates longer delay than CLUSTER-RW, it does not mean that the delay generated by CLUSTER is not optimal with *both* area and pin constraints. The pin constraint may have caused the longer delay. On average, the delay generated by CLUSTER increases by 4.3% compared with the optimal delay for pin constraint only generated by CLUSTER-RW. We also tested our CLUSTER program with different values in A, P, and D that are meaningful in multi-FPGA and multi-chip partitioning (e.g., we tested various values for P ranging from 20 to 150), and the results are comparable to the table shown here.

Allowing gate duplication enables our algorithm to achieve the best possible delay. However, a potential drawback of gate duplication is that it might lead to the usage of too many chips. In order to determine the effects of gate duplication on the number of clusters in the solution, we implemented a partitioning (without gate duplication) algorithm based on the classic FM heuristic, called Area/Pin-FM, where each partition must satisfy both area and pin constraints, but the delay information is ignored. We set $A = 100$ and $P = 80$, which resembles the constraints for Xilinx 3030 series. The results are presented in Table 2. It turns out that the delay generated by CLUSTER is much better than that by Area/Pin-FM as expected. But surprisingly, CLUSTER (with post-processing) even uses comparable or fewer chips than Area/Pin-FM does. On average, Area/Pin-FM generates solutions with 32.4% longer delay and 2.1% more chips than CLUSTER does.

The results demonstrate that allowing gate duplication does not mean that more chips will be used. This is due to the fact that logic resources in chips are often not fully utilized because of the limited pin constraint. The results confirmed our belief that gate duplication is an effective method to improve delay without using more chips for pin-constrained clustering.

The running times of most examples are less than 1 minute on a Sun SPARC 10 workstation. A couple of large circuits take 15-30 minutes, with more than half of the time spent on the post-processing step.

7 Concluding Remarks

In this paper, we have presented a circuit clustering algorithm that minimizes circuit delay subject to both area and pin constraints on each chip, using the general delay model. Our algorithm is optimal under either the area or the pin constraint only. Under both constraints, our algorithm achieves optimal delay in most cases. We have outlined the condition in which the non-optimality occurs and the experimental results show that the condition occurs rarely in practice. We also compared our algorithm with an FM heuristic. Our solutions achieve significant improvement in delay and use comparable number of chips. The results confirmed our belief that gate duplication is an effective method to improve delay without increasing the number of chips for pin-constrained clustering.

We assumed that the input circuits to CLUSTER are combinational in this paper. However, our algorithm can be easily extended to minimize the clock cycle in synchronous sequential circuits, using a technique described in [MBSV91]. Given a synchronous sequential circuit, we can first break the feedback loops at the inputs of all edge-triggered flip-flops, then apply CLUSTER (with post-processing) to minimize the longest path delay between any pair of flip-flops, and finally add the feedback loops.

References

[BRSV87] R. K. Brayton, R. Rudell, and A. L. Sangiovanni-Vincentelli. MIS: A Multiple-Level Logic Optimization. *IEEE Trans. on CAD*, pages 1061–1081, Nov. 1987.

[CD92] J. Cong and Y. Ding. An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 48–53, Nov. 1992.

[Eve79] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 175–181, 1982.

[LLT66] E. L. Lawler, K. N. Levitt, and J. Turner. Module Clustering to Minimize Delay in Digital Networks. *IEEE Trans. on Computers*, C-18, 1966.

[MBSV91] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. On Clustering for Minimum Delay/Area. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 6–9, 1991.

[PL88] B. Preas and M. Lorenzetti. *Physical Design Automation of VLSI Systems*. Benjamin/Cummings, 1988.

[RW93] R. Rajaraman and D. F. Wong. Optimal Clustering for Delay Minimization. In *Proc. 30th ACM/IEEE Design Automation Conf.*, pages 309–314, 1993.